Design of a Broadcast Packet Switching Network

Jonathan S. Turner
jst@wucs.UUCP
WUCS-85-4
March 1985

ABSTRACT

This paper describes a high performance packet switching network that can be used to provide voice, data and video communication on a large scale. A novel feature of the system is its flexible broadcast capability, which makes it suitable for a wide range of applications, including commercial television distribution and conferencing. The basic switching capability is provided by a high performance packet switch, called a Switch Module (SM), which terminates up to 63 fiber optic communications links (FOL), operating at a speed of 100 Mbs. The SM is designed as a modular component that can be used to construct systems ranging from a medium-sized business system to a large packet switching exchange, capable of providing service to over 200,000 users.

The research described here was supported by a grant from Bell Communications Research.

Design of a Broadcast Packet Switching Network

Jonathan S. Turner
Washington University
jst@wucs.UUCP

1. Introduction

This paper describes a high performance packet switching network that can be used to provide voice, data and video communication on a large scale. A novel feature of the system is its flexible broadcast capability, which makes it suitable for a wide range of applications, including commercial television distribution and conferencing. The basic switching capability is provided by a high performance packet switch, called a *Switch Module* (SM), which terminates up to 63 fiber optic communications links (FOL), operating at a speed of 100 Mbs. The SM is designed as a modular component that can be used to construct systems ranging from a medium-sized business system to a large packet switching exchange, capable of providing service to over 200,000 users.

This research is an outgrowth of earlier work that the author and colleagues did on the design of large-scale packet switching systems for voice and data, at Bell Laboratories. Details of that earlier work can be found in references [3], [4], [5] and [6]–[?]. The main novelty of the current work is its focus on higher bandwidths and broadcast.

The major components of the network are identified in Figure 1. Access to the network is provided through *Network Interfaces* (NI), which provide concentration, network protection and accounting functions. Each NI is connected to a *Packet Switch* (PS) by 30 *Fiber Optic Links* (FOL) and connects to 300 customers. The Packet Switches range in size from small systems terminating 1800 FOLs and capable of supporting 15,000 customers to large packet switching exchanges terminating 27,000 FOLs and capable of supporting 200,000 customers.

The network provides three basic communications services.

- *Point-to-point channels.* These are two-way channels joining pairs of users. The user re-

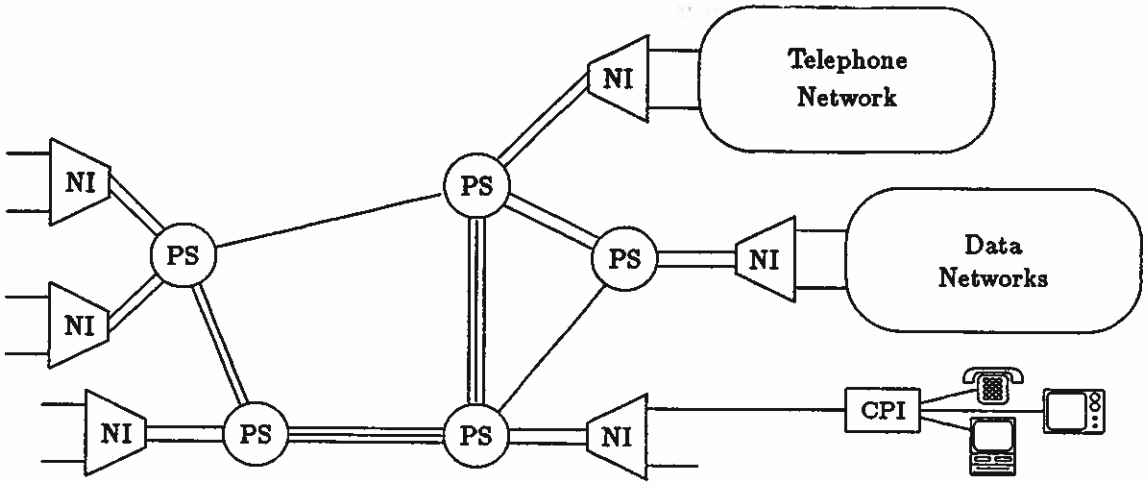


Figure 1: Network Architecture

quests a channel capable of providing a certain average bandwidth and the network allocates the requested bandwidth to the connection. If the requested bandwidth is not available, the connection is blocked. These connections do not provide flow control or error correction.

- *Datagrams.* These are individual packets, not associated with a pre-established connection. The network makes an effort to deliver them, but does not guarantee delivery.
- *Broadcast channels.* Any user can set-up a broadcast source that other users can then connect to. The average bandwidth of the source must be specified when it is established, and can range from a few bits per second to over a gigabit per second. There is no limit on the number of users that can receive a given broadcast signal. Thus, it is suitable for a variety of applications including commercial television and voice or video conferencing.

Figure 2 illustrates the way broadcast channels work. Two broadcast sources are shown at the top of the figure. At various points in the network, the signals are split into multiple copies, which are ultimately delivered to the approprate users. Thus, each broadcast source and its associated connections induce a tree in the network. When a user disconnects from a broadcast channel, the corresponding branch of the tree is removed. When a user requests connection to a channel, the network attempts to find a nearby branch of the broadcast tree and connects the user at the nearest point. Thus, the broadcast trees grow and shrink as usage patterns change, but for widely distributed channels, one can expect the bulk of the activity to be concentrated near the leaves.

A special-purpose switching fabric is used to support the broadcast service. This switch fabric

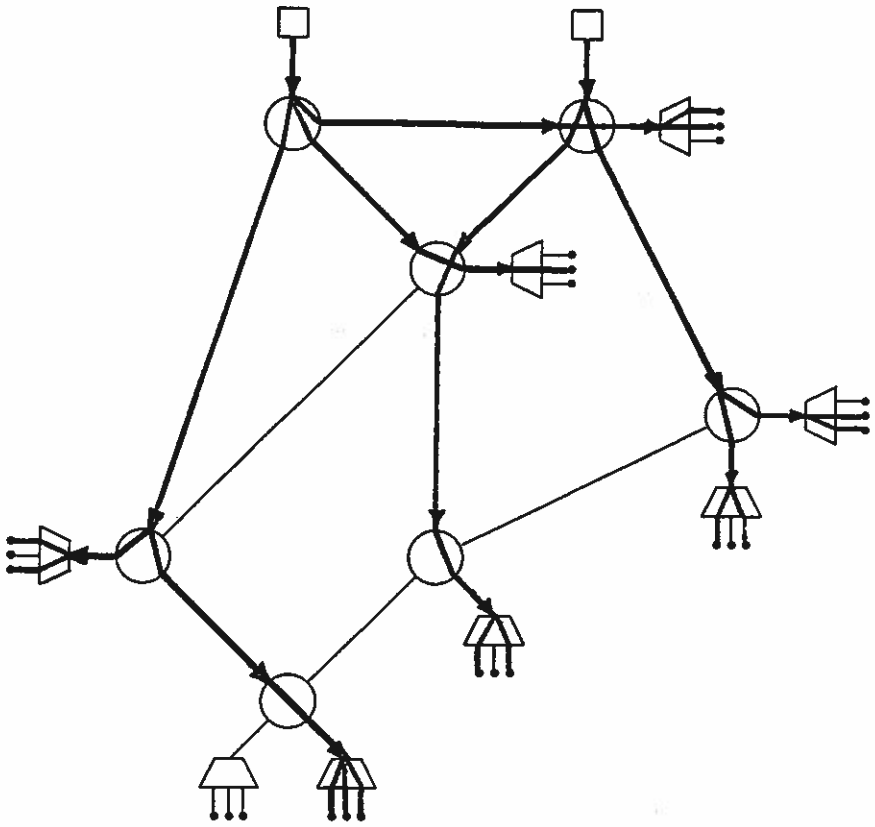


Figure 2: Broadcast Trees

splits a broadcast channel into multiple parts as needed. The switching fabric also allows a set of communications links to be treated as a group with traffic distributed over all links in the group. This makes it possible for the network to provide communications channels that require more bandwidth than is available on a single FOL.

The network is predominantly connection-oriented, allowing bandwidth allocation on a per connection basis. This means that the network can guarantee a certain average bandwidth for the duration of a connection, ensuring users adequate performance for their needs. Congestion control is provided by three mechanisms. The primary one is connection blocking—new connections are refused if adequate bandwidth is not available. An enforcement mechanisme is provided at the NIs, to ensure that users don't exceed their requested bandwidth for extended periods of time. The second mechanism is a system of user-specified packet priorities. During periods of heavy load, low priority packets are preferentially discarded to reduce congestion. This mechanism can be very

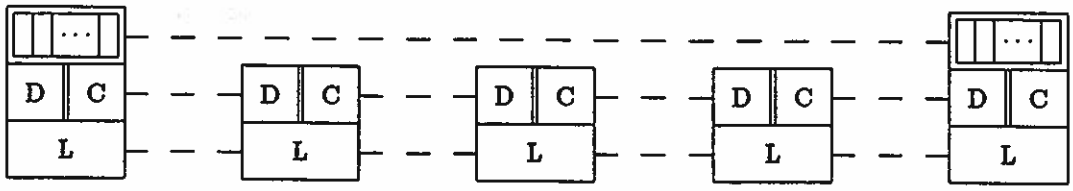


Figure 3: Protocol Structure

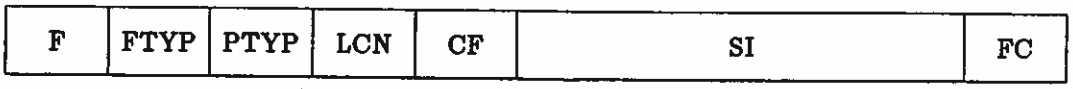
effective in a network dominated by video traffic, which can tolerate occasional information loss. The third mechanism is a congestion control field within each packet which can be modified by the network to inform the users of internal congestion. Users are expected to respond to this by temporarily slowing down their rate of transmission. Enforcement is provided at the NIs. Note that the first congestion control mechanism operates on a fairly long time-scale, the second on a very short time-scale and the third on an intermediate time-scale.

The communications protocols are divided into three levels—the *link level*, the *network level* and the *customer level*. The interrelationships among the different levels are indicated in Figure 3. There is a single link level protocol, two network level protocols (one for connection-based communication and one for datagrams) and a variety of customer level protocols. The customer level protocols are not discussed in detail here, but the intention is that these protocols would be application-dependent and built on top of one of the two network level protocols. One of these would be a telephony protocol. Another might be an internet protocol such as the DARPA IP to facilitate communication among different networks. Still another might be a connection-oriented internet protocol.

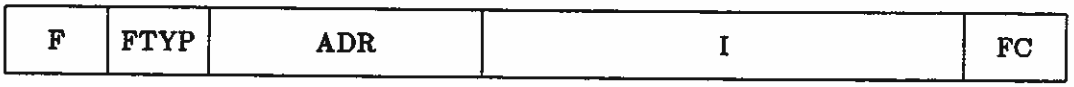
There are essentially three packet formats of interest to the network—*connection control packets*, *data transfer packets* and *datagram packets*. These packet formats are illustrated in Figure 4.

Link Level Protocol. The link level protocol provides frame delimiting, link transparency, error detection, and congestion control, but not error correction. There are two fields used by the link level protocol, the Frame Type field (FTYP) and the Frame Check field (FC). The FTYP field is one byte long and identifies the frame as a test frame, a datagram or a frame belonging to a connection. It also has a subfield containing a customer-specified priority. The network preferentially discards low priority frames when internal network buffers are in danger of overflowing. The Frame Check field (FC) uses a two byte cyclic redundancy code to detect errors in the frame. Frames with errors are simply discarded. Frames are separated by Flags (F). Bit stuffing is not used—the flag may

(a) Data Transfer Packet Format



(b) Connection Control Packet Format



(c) Datagram Packet Format

Figure 4: Packet Formats

appear within the body of the packet. Since the packet length is fixed, frame synchronization is straightforward.

Network Level Protocols. There are two protocols at the network level, the Simple Datagram protocol (SDP) and the Simple Connection protocol (SCP). SCP packets contain a one byte Control field (C) and a two byte Logical Channel Number field (LCN). The Packet Type field (PTYP) identifies each packet as either a data packet or a control packet and contains a Congestion Control subfield, used to inform the NIs and customers of internal network congestion. The Logical Channel Number field identifies which connection a packet belongs to. SCP data packets also contain an Information field, which can have any length up to 594 bytes. SCP control packets are used to establish and control connections and contain a Control Function field (CF) and a Supplementary Information field (SI). SDP packets contain an eight byte address field. Two bytes are reserved as a socket number, to be interpreted by the customer protocols. The remaining six bytes define a hierarchical address space organized on a geographical basis, to permit routing of connections and datagrams based on local knowledge of network topology.

The remainder of this paper provides a detailed description of the components that make up the network. Section 2 describes the design of the Switch Modules. Section 3 shows how the SMs can be used to construct Network Interfaces and Packet Switches. Section 4 offers some conclusions and outlines extensions to the SM that provide additional capablities.

2. Design of the Switch Module

The overall structure of the Switch Module (SM) is shown in Figure 5. The SM terminates up

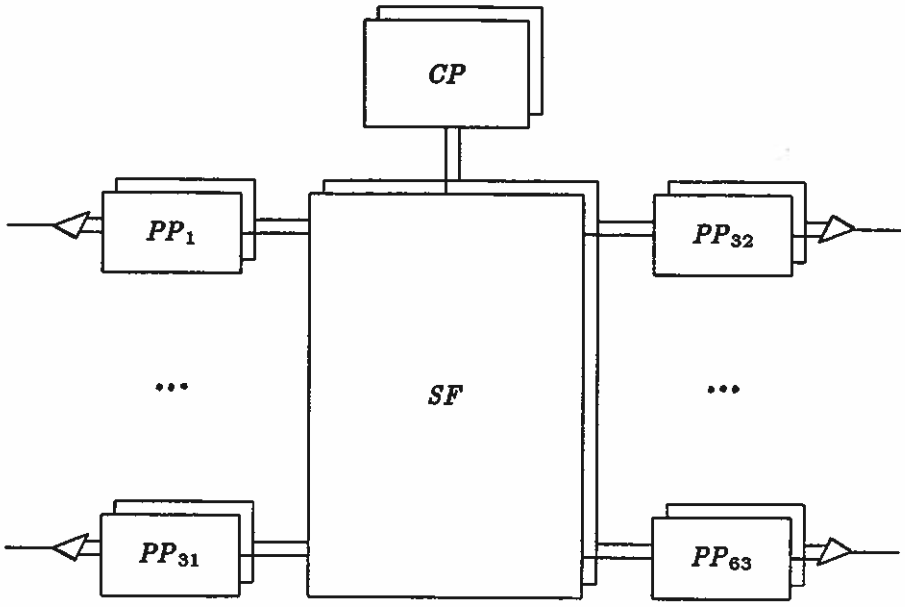


Figure 5: Switch Module

to 63 FOLs, engineered for a maximum occupancy of 80%, giving the SM a raw throughput of over 5 Gbs, or about 1,200,000 packets per second.

The *Packet Processors* (PP) perform the link level protocol functions, including the determination of how each packet is routed. This includes routing connection control packets to the Connection Processor and datagram packets to one of several Datagram Routers (not shown).

The *Switching Fabric* (SF) is the heart of the SM. It has a highly parallel organization and provides multiple paths for packets to ensure good performance. The SF also replicates broadcast packets as necessary and can support distribution of packets across a set of outgoing FOLs.

The *Connection Processor* (CP), is responsible for establishing connections, including both point-to-point and broadcast connections. To do this, it exchanges control packets with CPs in neighboring SMs and controls the actions of the PPs and SF by writing information in their internal control tables. It also performs a variety of administrative and maintenance functions.

The *Datagram Routers* (DR), which are not shown in the figure, are special-purpose devices used to route datagrams. The number of DRs can be engineered to suit the traffic. Each one occupies a port on the SF, replacing one PP. Since the vast majority of the traffic is expected to be connection-oriented, a few DRs (≤ 5) should suffice for most situations.

The duplication scheme shown in the figure is designed to ensure the highly reliable operation required in a large scale public communications network. The normal mode of operation is for one plane to be active with the other acting as a "hot standby." A memory update channel is provided between the two CPs so that the memory of the standby CP tracks that of the active one. The standby CP executes an audit program designed to ensure that the tables in the PPs on the standby side also track those in the active side. Optical switches control which plane is connected to the FOLs. A system reconfiguration would normally cause a loss of all packets in the active switch plane before reconfiguration, but would cause no lost connections.

When a packet enters the SM, it is reformatted by the addition of three new fields, the Routing field, the Control field and the Source field. The *Control* field (C) is one byte long and identifies different packet types within an SM, including various sorts of control packets. The *Source* field (SF) is one byte long and identifies the origin of the packet within the SM. The Routing field (RF) is four bytes long and contains information needed to route the packet through the switch fabric. It has three subfields. The Routing Control subfield (RC) determines the type of routing the packet is to receive—the most important distinction is between broadcast packets and others. The interpretation of the remaining two subfields depends on the RC value. For packets belonging to point-to-point connections, the RF contains a *Group Number* subfield (GN) and a *Logical Channel Number* subfield (LCN). To simplify the presentation, a complete discussion of group numbers will be deferred until later—for the present it will suffice to identfy group numbers with link numbers. The LCN subfield contains the LCN that the packet will carry when it leaves the SM. For broadcast packets, the RF contains a *Number-of-Copies* subfield (NC) and a *Broadcast Channel Number* (BCN). The NC subfield is equal to the number of outgoing FOLs that require a copy of this broadcast channel. The BCN is used to distinguish this channel from all other broadcast channels within the SM.

Logical channel translation is the process used to determine how to route a packet belonging to a connection through the SM. Each PP contains a *Logical Channel Translation Table* (LCXT) used for this purpose. When a packet is received by a PP, its LCN is used to index the LCXT. The selected entry is copied into the Routing Field of the packet. The LCXT contains 4096 entries of four bytes each. The entries in the LCXTs are maintained by the CP, which typically writes a new entry each time a new connection is established.

2.1. Packet Processor

The structure of the Packet Processor is shown in Figure 6. It contains four packet buffers.

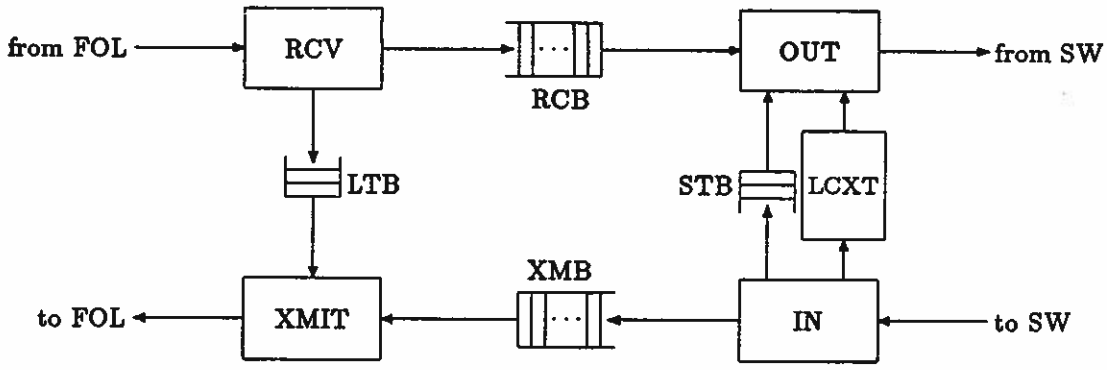


Figure 6: Packet Processor

The *Recieve Buffer* (RCB) is used for packets arriving from the FOL and waiting to pass through the SF. The *Transmit Buffer* (XMB) is used for packets arriving from the SF that are to be sent out on the FOL. The *Link Test Buffer* (LTB) and *Switch Test Buffer* (STB) provide paths for test packets used to verify the operation of the FOL and SF respectively. The RCB has a capacity of 16 packets, the XMB has a capacity of 32. The LTB and STB can each hold two packets. The four buffers require a total of about 240 Kbits of memory.

The *Logical Channel Translation Table* (LCXT) is a dual port memory that implements a table with 4096 entries of four bytes each. It can be read by the Output Circuit and written by the Input Circuit.

The *Receive Circuit* (RCV) converts the incoming optical signal to an electical signal in eight bit parallel format, synchronizes it to the local clock, discards packets with header errors, routes test packets to the LTB and other packets to the RCB.

The *Output Circuit* (OUT) takes packets from the RCB, performs the logical channel translation described above and sends the packets on to the switch. It also processes test packets from the STB.

The *Input Circuit* (IN) routes data packets to the XMB, removing the RF, SPP and C fields in the process. It also routes switch test packets to the STB, and updates the LCXT memory in response to LCXT write packets.

The *Transmit Circuit* (XMIT) takes packets from the XMB, adds the flag field and converts from the eight bit parallel format to an optical signal. It also processes test packets from the LTB.

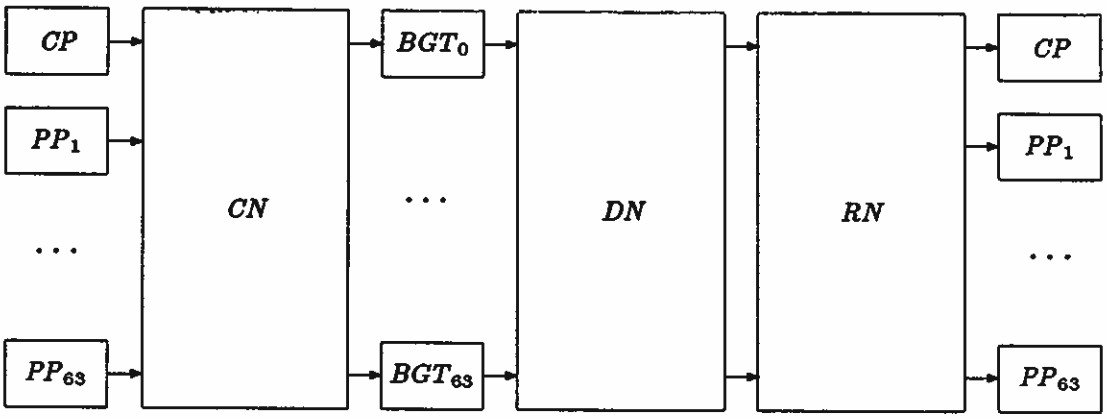


Figure 7: Switch Fabric

2.2. Switch Fabric

A block diagram of the Switch Fabric (SF) is given in Figure 7. It contains four major components, a *Copy Network*, a set of *Broadcast and Group Translators*, a *Distribution Network* and a *Routing Network*. The SF runs at a clock rate of 25 Mbs and has eight bit wide internal data paths. This gives an effective bit rate of 200 Mbs on the internal data paths, roughly two times the speed of the FOLs. An occupancy of 80% on the FOLs translates to a 40% occupancy on the internal data paths, which keeps contention and delay low.

When a broadcast packet having k destinations passes through the Copy Network (CN), it is replicated so that k copies of that packet emerge from the CN. Point-to-point packets pass through the CN without change.

The Broadcast and Group Translators (BGT) perform two translation functions. First, for each arriving broadcast packet they determine the proper outgoing GN (group number) and LCN. Then, they translate the GN to an LN (link number). Discussion of this latter translation is postponed until later in this section. For the moment, we will assume that the GNs and LNs are identical.

The Distribution and Routing Networks (DN,RN) move the packets to the proper outgoing PP. The RN is a straightforward binary routing network and the DN is provided to prevent internal congestion within the RN.

Routing Network and Distribution Network

The RN is a 64×64 binary routing network. Its structure is illustrated in Figure 8 which shows

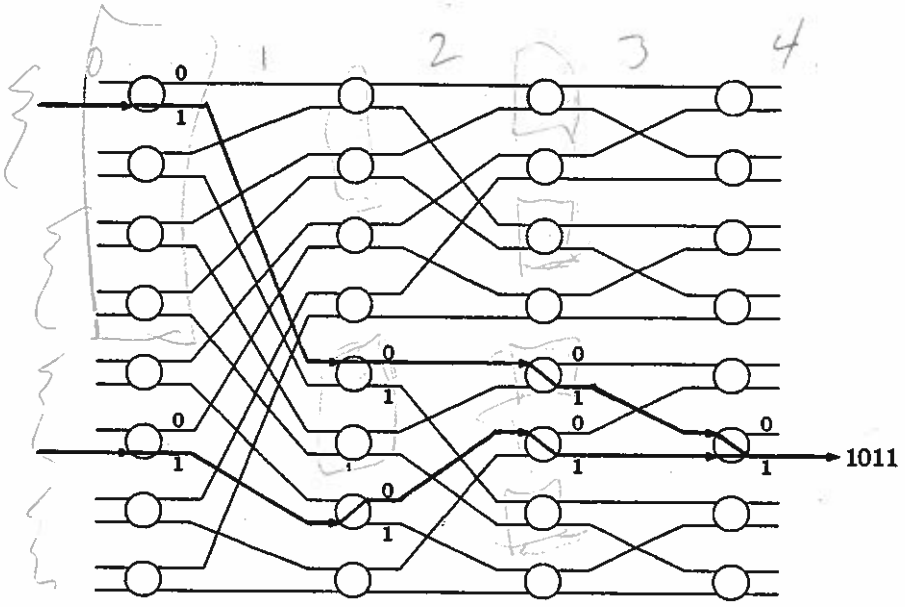


Figure 8: 16 × 16 Binary Routing Network

a 16 × 16 version. The key property of such networks is that they are *bit addressable*—that is, the route a packet takes through the network is determined by successive bits of its destination address. The figure shows paths from two different input ports to output port 1011. Note that at the first stage the packet is routed out the lower port of the node (corresponding to the first one bit of the destination address), at the second stage it is routed out the upper port (corresponding to the zero bit) and in the third and fourth stages it is routed out the lower ports. The self-routing property is shared by a variety of different although similar interconnection patterns, including the so-called delta, shuffle-exchange and banyan networks [2].

Buffers are provided at the inputs of each node. Each buffer is capable of holding two complete packets, implying about 20,000 bits of memory per node. A packet may pass through a node without being buffered at all if the desired output port is available when the packet first arrives. Indeed, in a lightly loaded network, a packet can pass through the RN with no buffering at all. In this case it encounters a delay of just a few clock times in each node.

The data paths between nodes are eight bits wide. In addition, there is a single upstream control lead used to implement a simple flow control mechanism. This prevents loss of packets due to buffer overflows within the fabric. The entire network is operated synchronously, both on a bit basis and a packet basis—that is, all packets entering the first stage do so during the same clock cycle. The clock rate is 25 Mbs, giving an effective bandwidth of 200 Mbs for each of the paths

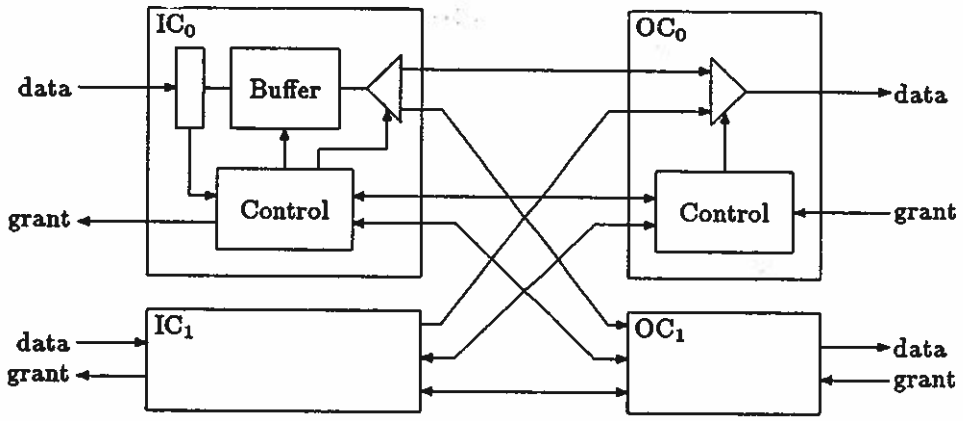


Figure 9: Switch Node

through the network.

Figure 9 is a more detailed picture of a single switch node. The switch node consists of two *Input Controllers* (IC) and two *Output Controllers* (OC). The ICs determine the routing required for each packet, request the appropriate OC, buffer packets if necessary, and apply upstream flow control to prevent buffer overflow. The Output Controllers (OC) arbitrate requests for the outgoing links.

One problem with binary routing networks is that they can become congested in the presence of certain traffic patterns. This is illustrated in Figure 10, which shows a traffic pattern corresponding to several *communities of interest.* In this pattern, all traffic entering the first four inputs is destined for the first four outputs, all traffic entering the second group of four inputs is destined for the second group of four outputs, and so forth. Note that with this pattern, only one fourth of the links joining the second and third stages are carrying traffic. Thus, if the inputs are heavily loaded the internal links will be hopelessly overloaded and traffic will back up. In a 64×64 network, there are eight stages and the links between the third and fourth stages can in the worst-case be carrying all the traffic on just 8 of the 64 links.

The DN solves this problem by evenly distributing packets it receives across all its outputs. It has the same internal structure as the RN, but its nodes ignore the destination addresses on packets and route them alternately to each of their output ports. This strategy is modified if one or both ports is unavailable. In this case, the first port to become available is used. This approach breaks up any communities of interest and makes the combination of the DN and RN robust in the face of

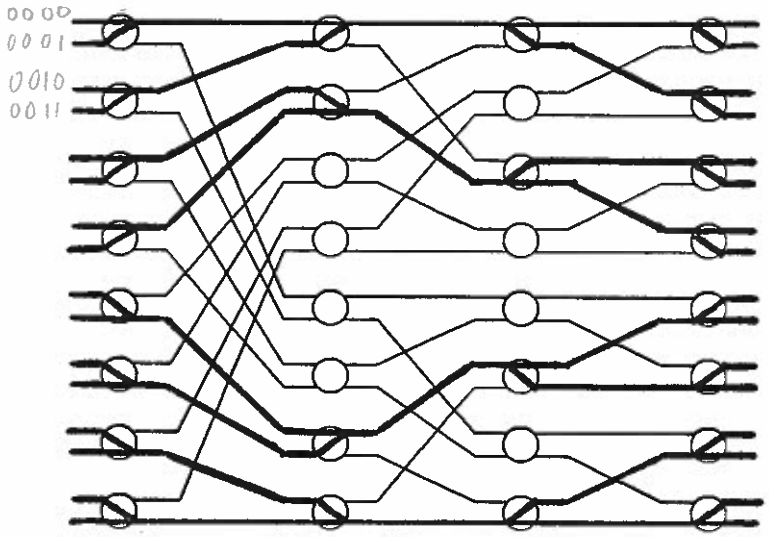


Figure 10: Congestion in Binary Routing Networks

pathological traffic patterns.

Copy Network and Broadcast Translation

The structure of the Copy Network (CN) is the same as that of the RN and CN. The CN's function is to make copies of broadcast packets as they pass through. This is illustrated in Figure 11. The packet entering at left belongs to broadcast channel 36 and is to be sent to seven different links. At the first stage, the packet is routed out the upper port. This is an arbitrary decision—the lower link could have been used at this point. At the second stage, the packet is sent out on both outgoing links and the number of copies fields in the outgoing packets is modified. The upper packet will generate four copies and the lower one three.

We now describe the CN routing algorithm for broadcast packets. Let *BCN* and *NC* be the broadcast channel field and the number-of-copies field from the packet and let *sn* be the stage number of the node in the switch, where stages are numbered from right to left, starting with 1.

- If $NC > 2^{sn-1}$ request both output links and when the links are available, simultaneously send the packet out on both.
  - If *BCN* is even, the *NC* field of the upper packet is set to $\lfloor (NC+1)/2 \rfloor$ and the *NC* field of the lower packet is set to $\lfloor NC/2 \rfloor$.

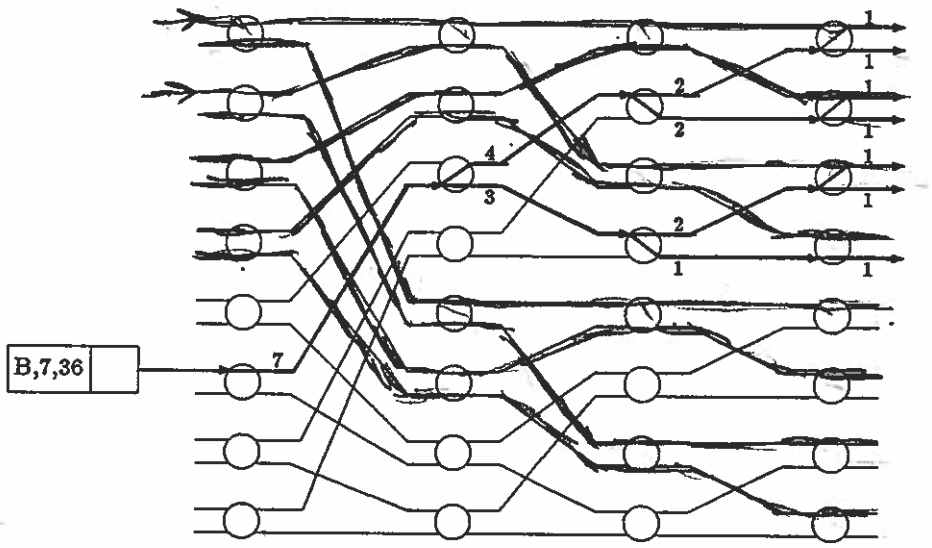


Figure 11: 16×16 Copy Network

- If BCN is odd, the NC field of the upper packet is set to $\lfloor NC/2 \rfloor$ and the NC field of the lower packet is set to $\lfloor (NC+1)/2 \rfloor$.

- If $NC \leq 2^{sn-1}$, use the DN algorithm.

Note that this algorithm delays splitting packets as long as possible. Another option is to split the packets early. However, this approach can lead to congestion in the CN. The late-splitting algorithm avoids this problem.

An important property of the routing algorithm is that given a particular combination of BCN and NC, we can predict which of the NC copies can appear at each output of the copy network. The index of the copy that appears at a particular output port is called the *broadcast copy index* or *bci* for that port. Note that the *bci* for a particular output port is a function of the NC field and the low order bit of the BCN. We use the notation $bci_k(NC, BCN)$ to denote the *bci* of port k for a particular combination of NC and BCN.

A simple algorithm for computing the *bci* function is shown in Figure 12. Actually, this program computes a complete binary tree—when it is done, the *bci* field of the k^{th} leaf node is equal to $bci_k(NC, BCN)$. An example of a tree computed by the algorithm is shown in Figure 13.

When broadcast packets emerge from the CN, their final destination is yet to be determined.

```
tree_node bci_tree(integer i, NC, BCN, sn);
    tree_node t;
    t := new(tree_node);
    if sn = 0 →
        t → bci := i;
    | sn > 0 ∧ NC ≤ 2^(sn-1) →
        t → left := bci_tree(i, NC, BCN, sn − 1);
        t → right := bci_tree(i, NC, BCN, sn − 1);
    | sn > 0 ∧ NC > 2^(sn-1) ∧ even(BCN) →
        t → left := bci_tree(i, ⌊(NC + 1)/2⌋, BCN, sn − 1);
        t → right := bci_tree(i + ⌊(NC + 1)/2⌋, ⌊NC/2⌋, BCN, sn − 1);
    | sn > 0 ∧ NC > 2^(sn-1) ∧ odd(BCN) →
        t → left := bci_tree(i, ⌊NC/2⌋, BCN, sn − 1);
        t → right := bci_tree(i + ⌊NC/2⌋, ⌊(NC + 1)/2⌋, BCN, sn − 1);
    fi;
    return(t);
end;
```

Figure 12: Algorithm for Broadcast Copy Index Computation

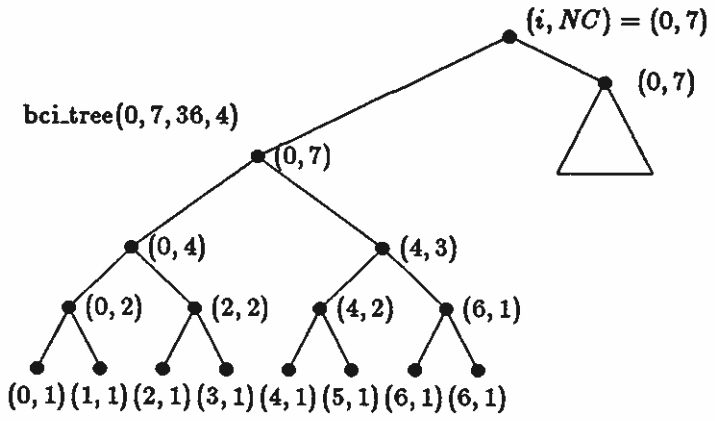


Figure 13: Computation of Broadcast Copy Indices

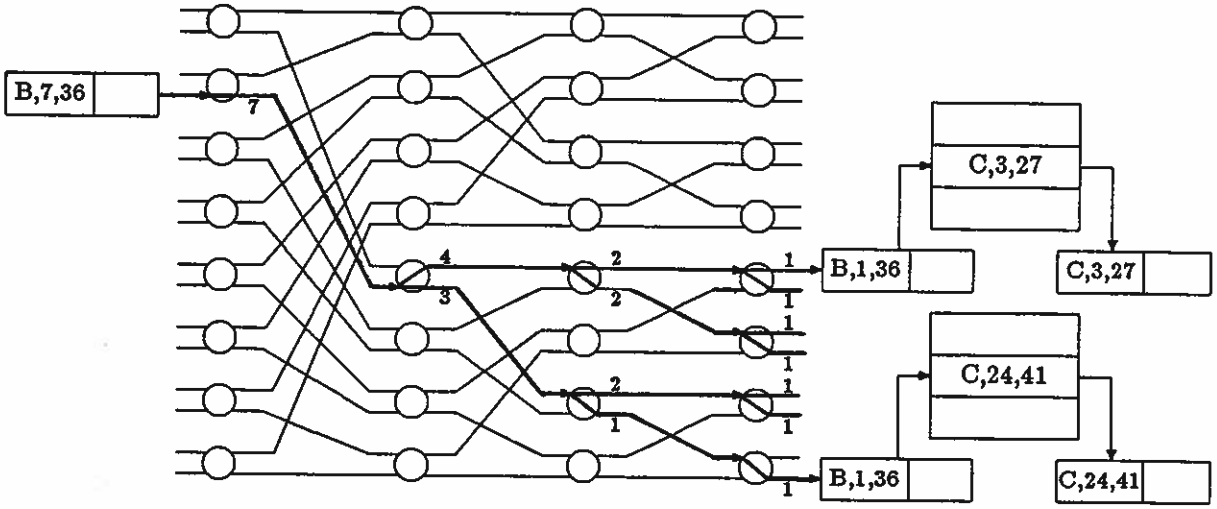


Figure 14: Example of Broadcast Processing

It is the function of the Broadcast and Group Translator (BGT) to determine where each broadcast packet should go. Each BGT contains a *Broadcast Translation Table* (BTT), used for this purpose. The BTT is indexed by the BCN of the incoming packet. The selected entry contains the appropriate outgoing group number (which we're still assuming to be the same as the link number) and LCN for the packet, and is copied to the packet's RF field.

Note that two BGTs need not have identical entries in their BTTs for a given BCN, since their *bci* values may be different. This point is illustrated in Figure 14 which shows the translation process for several packets copied from a single broadcast packet.

The addition of a new destination to a broadcast channel can radically change the mapping required in the BTTs. In general, when a new destination is added to a particular broadcast channel, the CP must compute a new broadcast copy index tree as described above and use that information to update all the BTTs. Since this can happen frequently, it appears likely to present a substantial computational burden for the CP. Fortunately, there is a simple solution to the problem. It requires that for $0 \leq k \leq 63$, BGT_k contains a table that it can use to compute $bci_k(NC, BCN)$. Since, $NC \leq 64$ and only the least significant bit of the *BCN* is needed to compute *bci*, the table has 128 entries, each one byte long. Note that this table is static and is different for each BGT. Now, when the CP wishes to update the BTTs for a particular broadcast channel, it sends a control packet of the form shown in Figure 15.

RF	C		I
B,64,36	BGT_update		$NC, (GN_0, BCN_0), (GN_1, BCN_1), \ldots, (GN_{NC-1}, BCN_{NC-1})$

Figure 15: Format of BTT Update Packet

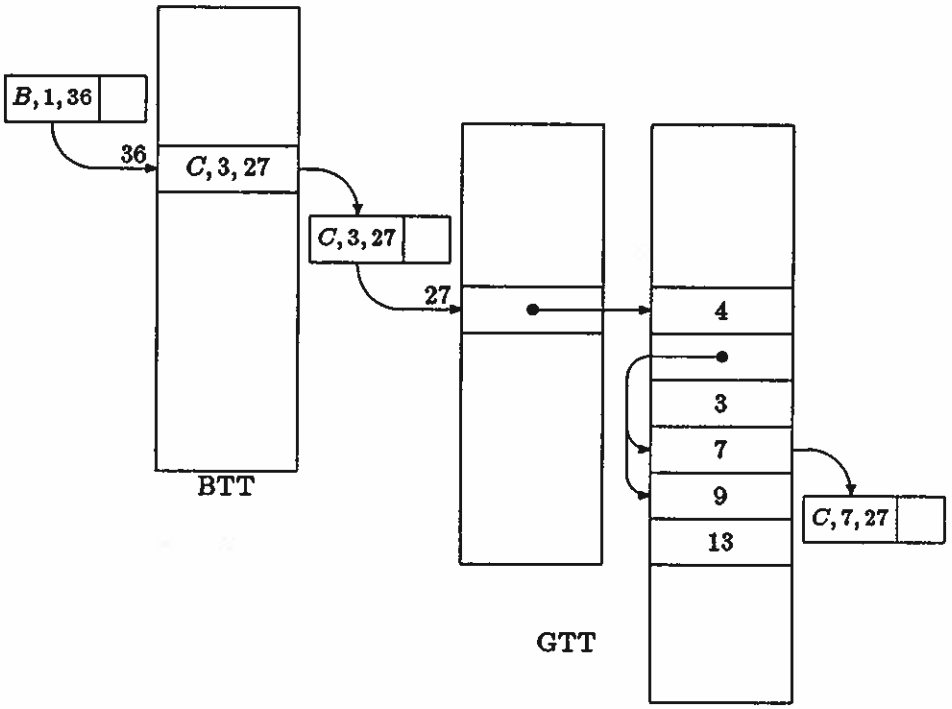


Figure 16: Broadcast and Group Translation

The CN replicates the packet so that each BGT receives a copy. When BGT_k receives its copy, it extracts NC and BCN from the packet, then uses its lookup table to compute $j = bci_k(NC, BCN)$ and finally copies (GN_j, LCN_j) from the packet to $BTT_k[BCN]$.

Using this scheme, the CP keeps a copy of the *BTT_update* packet in its memory. To add a new destination, it increments the NC field, adds a new (GN,LCN) pair to the end of the packet and sends it out to the SF. To remove a destination, it decrements the NC field, removes the proper (GN,LCN) pair from the packet and sends it out.

Link Groups

A small addition to the mechanisms described above, allows links to be collected together into *link groups*. The links in a group must have the same destination. Traffic for a link group is evenly

distributed over all of its links. This allows dynamic load distribution across a set of links permitting higher utilization. It also allows the network to support channels that require more bandwidth than any one FOL can provide.

When a BGT receives a point-to-point packet, the packet contains a Group Number (GN) in its Routing Field. Each BGT contains a table, called the Group Translation Table (GTT), which it uses to translate a GN to a Link Number (LN). The GTT identifies the links in each group and also contains a pointer to a particular link in the group. Every time a packet for a particular group is handled, this pointer is advanced to the next link in the group, so that each BGT distributes traffic evenly across the group. Broadcast packets are handled similarly, but they must first go through the broadcast translation process described earlier. The organization of the GTT is shown in Figure 16, which illustrates the entire translation process for a broadcast packet. Note that the GTT is a fairly static table—it changes only when groups are reconfigured, not on a per-connection basis.

One consequence of link groups is that the LCXTs for all links in the same group must be identical. Consequently, on every connection the CP must update the LCXTs for all links in the group. This can be handled easily by sending a broadcast LCXT update packet.

2.3. Implementation Details

This section is an an attempt to 'size up' the Switch Modules. While the estimates given here are rough, they give a fairly accurate picture of the complexity of the SMs.

The switch nodes can best be implemented using custom integrated circuits. Each of the nodes that makes up the switch fabric consists of about 20,000 bits of memory plus some control circuitry. The memory is the dominant component, so we can estimate the size of a chip needed to implement a switch node by looking at the area required for the memory. The memory in the switch nodes is not extremely large, but it must be fast. Perhaps the simplest way to implement it is using a large dynamic shift register. A single bit of shift register memory can be implemented in $200\,\mu m^2$, assuming a $1\,\mu m$ CMOS process, implying that 20,000 bits of memory will consume about $4\,mm^2$. This suggests that several nodes can be implemented on a single chip. Not surprisingly, pin limitations restrict the size of the chip more severely than chip area. A single node on a chip requires 36 signal pins plus additional pins for power, ground and clock. Four nodes can be placed on a chip with about 80 pins if they are configured as a 4×4 switch. Actually, a better way to implement a 4×4 switch is to do it as a single larger node, rather four small nodes—this structure requires half as much memory and is about twice as fast. It is convenient to have two chip types—one chip

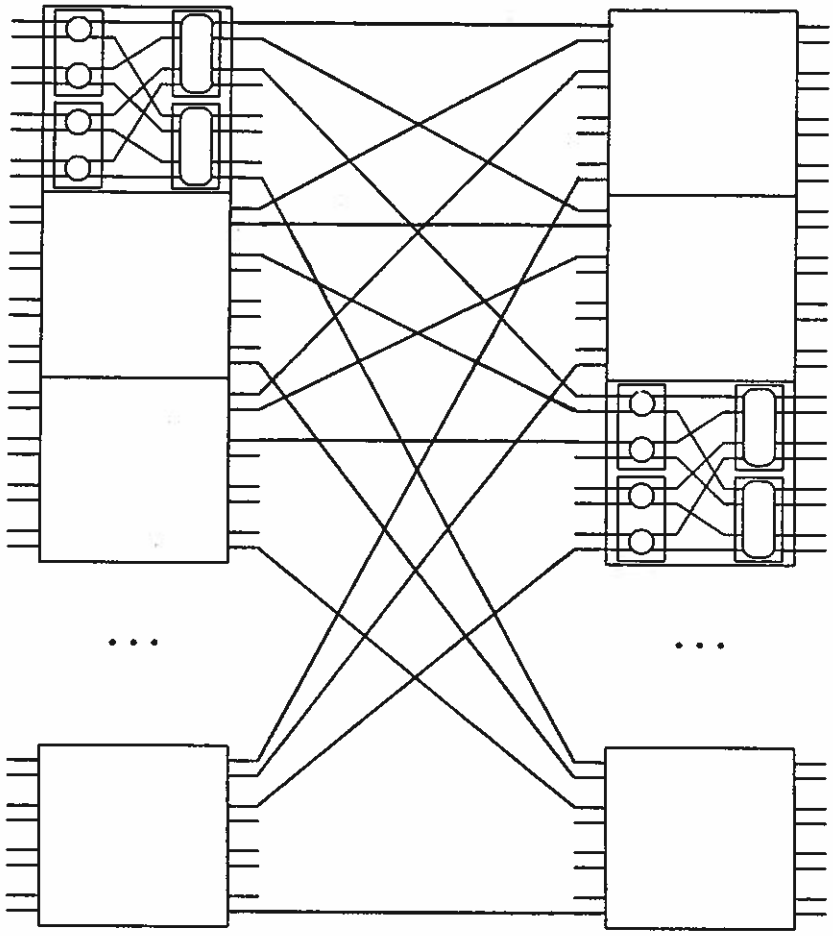


Figure 17: 64 × 64 Network

contains two independent 2 × 2 nodes and the other contains a single 4 × 4 node. Each of these can be implemented in an 80 pin package, preferably a high density package such as a pin grid array.

The packet processors are more complex than the switch nodes. Each requires about 400,000 bits of memory if one includes all the packet buffers and the logical channel translation table. The largest item is the transmit buffer, which can be implemented as eight 128 × 150 memory arrays. The cycle time of these arrays is not a critical performance factor—each access reads or writes 150 bits, leading to one access every 3 μs. Thus, fairly dense memory structures can be used. A three transistor memory cell can be implemented in about 100 μm^2 assuming a one micron technology, implying that the memory cells required by the transmit buffer will consume about 16 mm^2. The addition of the supporting circuitry required by the memory arrays could bring the total up to about 30 mm^2 The other packet buffers can be implemented in a similar fashion. Together, they contain

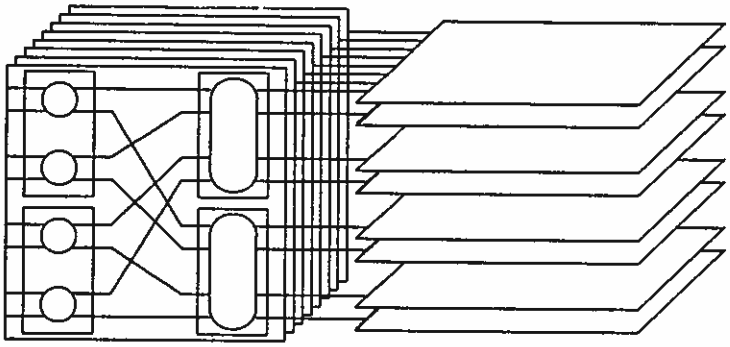


Figure 18: Arrangement of Circuit Cards for a Network

less than two thirds the memory of the transmit buffer, but we will estimate that they consume another 30 mm^2. The LCXT contains about 150,000 bits of memory organized in nine 128×128 arrays. This adds about another 30 mm^2 to the chip area. If we assume (conservatively) that the memory consumes 75% of the chip area required to implement the PP, we arrive at an estimate of $1.2 cm^2$ for the entire PP. A three chip implementation appear feasible, with the first chip containing the RCV, XMIT and STB blocks, the second containg the RCB and XMB and the third containing the IN, OUT, STB and LCXT blocks.

The Broadcast and Group Translators each contain several tables. The BCI table, used to compute a broadcast channel index contains 128 bytes, the BTT contains 4096 bytes (assuming 1024 unique BCNs in a SM), and the GTT 320 bytes. The total memory requirement is thus under 40,000 bits. Thus, two can fit comfortably on a single chip.

With these estimates in hand, we can consider how an entire SM might be put together. A convenient way to implement a 64×64 binary routing network is to subdivide it into 8×8 subnetworks as shown in Figure 17. Each of the 8×8 subnets can be constructed from four 2×2 nodes and two 4×4 nodes for a total of four chips, and can be implemented on a single board with 144 signal leads. These boards can then be arranged into two orthogonally positioned ranks to give a compact implementation of the 64×64 network. This structure is shown in Figure 18. What makes this arrangement particularly attractive is that the board-to-board signal paths can be kept very short.

An SM however, contains three such networks. We can employ the same idea in the SM using the arrangement shown in Figure 19. Here, there are four groups of eight circuit boards each. The boards at the bottom center contain the first half of the CN. The boards at the right contain the

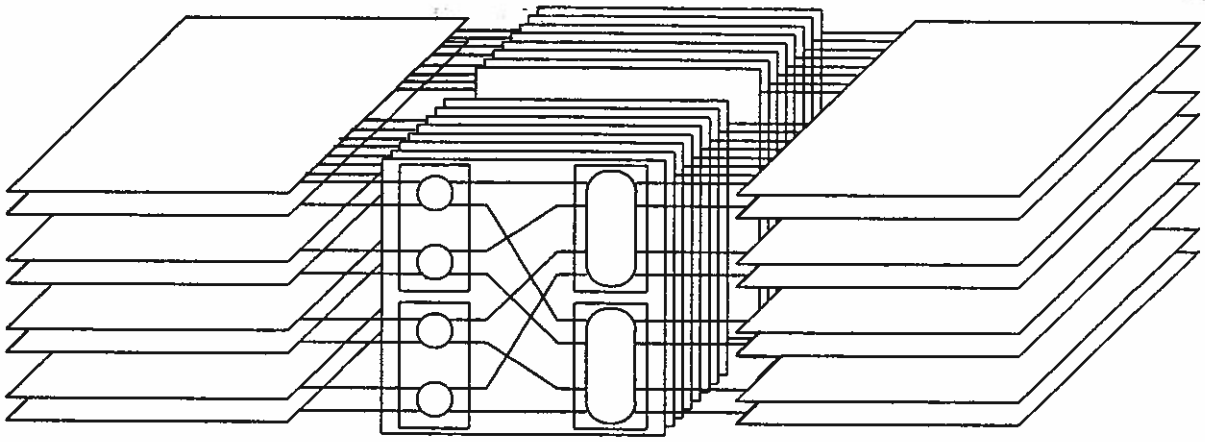


Figure 19: Switch Module Packaging

second half of the CN, the BGTs and the first half of the DN. The boards at the top center contain the second half of the DN and the first half of the RN. The board at the left contain the second half of the RN and the PPs. The most densely populated boards are the ones at the left, each of which contains four switch node chips and twenty-four PP chips. Assuming high density packaging in which chips are mounted in 1″ square pin grid arrays on multi-layer ceramic circuit boards, this entire assembly could fit in a volume of roughly 6″ × 12″ × 20″. A complete SM would contain two of these assemblies plus the CPs. Each CP contains a 32 bit microprocessor with perhaps 2 megabytes of memory, an interface to the switch fabric and a single terminal interface. Hence, it could be implemented on three or four circuit cards and four complete SMs could be packaged in a single 30″ wide cabinet.

3. Design of Network Components

The Switch Module is designed as a component that can be used to build larger systems. The decision to build large systems from small components can be criticized on the grounds that the total hardware required can be substantially larger and the performance lower than if a monolithic structure is used. There are several strong arguments in favor of the modular approach, (1) a system constructed out of small modules is easier to design and implement, (2) modular systems can grow over a large range of sizes by adding modules and (3) manuafacturing economics favor systems that contain a large number of identical components.

Modular systems are easier to design because the modules lead to a natural separation of

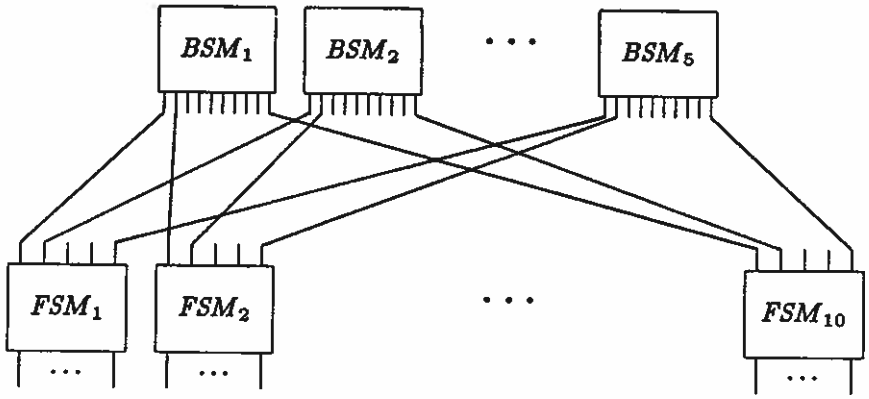


Figure 20: Network Interface Architecture

local concerns from global ones. Modules can be defined by their external interfaces and treated as 'black boxes' by their environment. Once this separation has been made explicit by a set of module specifications, the design of the individual modules is relatively straightforward. The size of the modules is central issue. In the case of the SMs, larger modules would have been preferable for constructing larger systems. The decision to use a fairly small module was motivated by three considerations. First, it allows a single processor of moderate proportions to handle all the connection processing, avoiding a number of difficult problems that must be faced in larger systems. Second, it allows a very simple approach to reliability and maintenance—to recover from a hard failure in the active side, you simply switch to the other side. With small enough modules, failure rates are low enough that there is no need to resort to more elaborate recovery algorithms. The small size also simplifies repair in the field—once it has been determined that a particular SM side is faulty, it can be replaced as a unit. Third, hardware timing is easier to handle in a system of modest physical dimensions. The SMs are small enough that it is reasonable to operate them as synchronous systems. Asynchronous operation need only be considered at the interface between the PPs and the FOLs.

The easy growability of modular systems is an important advantage, but can only be realized if the system structure provides some flexible 'glue' that can be used to connect the modules together. The glue used to construct large systems from SMs is called a Cross-Connect (XCON) and is discussed in some detail below.

Our first example of a larger system component constructed from SMs is the Network Interface (NI), shown in Figure 20. A single NI contains fifteen SMs, five of which are termed Back-end SMs (BSM) and ten of which are termed Front-end SMs (FSMs). Each FSM has six FOLs connecting it to each of the BSMs and each BSM has six FOLs connecting it to each of the FSMs. The NI has

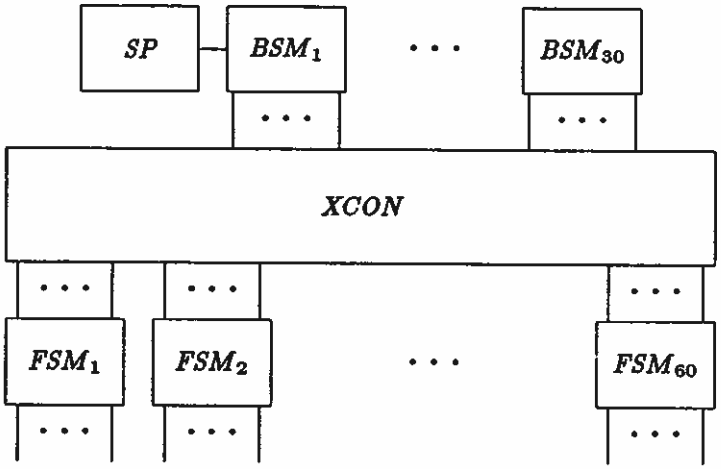


Figure 21: Small Packet Switch

330 FOLs that can be used to connect to other system components. Typically, 30 would be used to connect to the NI's host PS with the remainder going to customers. The acutal ratio of downstream links to upstream links is completely flexible—a ratio of about 10:1 appears reasonable, since most of the downstream bandwidth is likely to be consumed by a few popular broadcast channels.

SMs can also be used to build a Packet Switch (PS) as shown in Figure 21. The PS shown can have as few as one BSM and two FSMs or as many as 30 BSMs and 60 FSMs. In the largest configuration, it terminates 1800 FOLs on its FSMs. If 1500 FOLs are used to connect to NIs, such a PS can support 15,000 users. This leaves 300 FOLs for connecting to other PSs. Again, the ratio of downstream to upstream links is not limited by the architecture and can be adjusted to suit actual needs—the ratio chosen here appears reasonable. Each PS contains a Supervisory Processor (SP), which provides office administration and craft interface functions. The SP is a fairly large processor with its own disk sub-system and possibly other peripherals. It does no processing of individual connections, but does supervise the operation of the SMs through its control of their routing tables.

A PS must be capable of growing across a range of sizes. In the smallest configuration, it would have 30 FOLs joining each FSM to the one BSM. In the largest configuration, it would have one FOL joining each FSM to each BSM. If the SMs were joined directly to one another, growth of a PS would require manual recabling. This is avoided by the Cross-Connect (XCON), which can be thought of as an 1800×1800 cross-bar switch. Fortunately, the XCON doesn't require all the

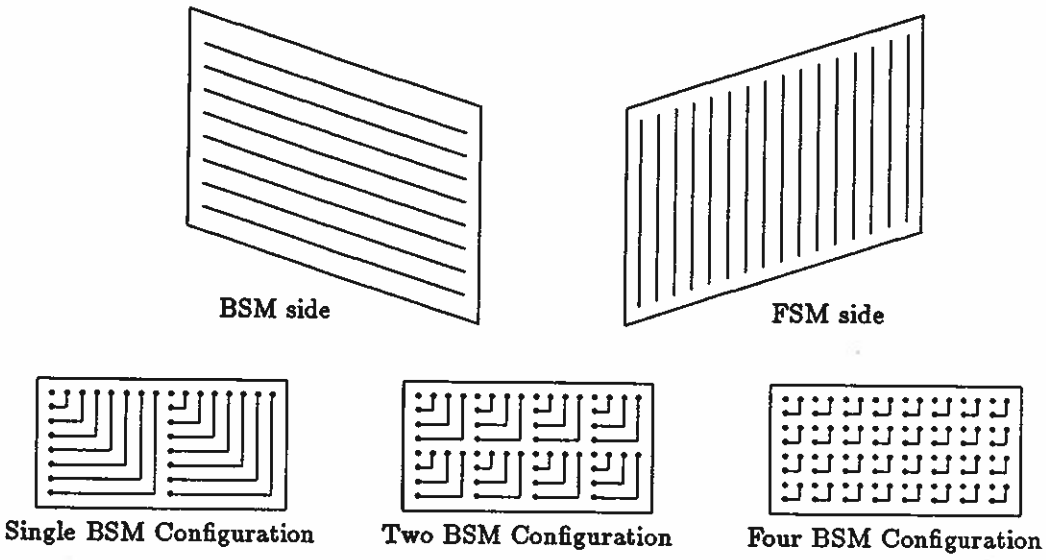


Figure 22: Cross Connect

functionality of a full cross-bar and can implemented in much less expensive way. The structure of the XCON is illustrated in Figure 22, which shows a small version. The XCON has a planar structure with a BSM side and an FSM side. The BSM side is organized in rows—all FOLs from a particular BSM terminate on the same row. The FSM side is organized in columns—all FOLs from a particular FSM terminate on the same column. The figure shows an XCON that is appropriate for SMs having just sixteen FOLs. Such an XCON can be used to construct a PS with up to eight BSMs and sixteen FSMs. In a fully grown configuration, all the rows on the BSM side are occupied, as are all the columns on the FSM side and the connections pass straight through the XCON—so each BSM/FSM pair is joined by one FOL. In the smallest configuration, the top row on the BSM side is occupied, along with the first and ninth columns on the FSM side. Switches in the plane of the XCON are set to provide eight paths connecting the BSM to each of the FSMs, as illustrated in Figure 22(a). Figure 22(b) shows a configuration in which two BSMs are connected to four FSMs, with four FOLs joining each pair and Figure 22(c) shows a configuration with four BSMs and eight FSMs. The complexity of an XCON grows linearly with the number of FOLs that connect to it, making it reasonable to construct systems with fairly large XCONs.

Larger packet switches can be constructed by combining modules in a similar way. A PS capable of terminating 27,000 FOLs can be constructed from 900 FSMs and fifteen of the smaller packet switches described above, which function as BSMs in that case. If 20,000 FOLs are used to connect to NIs, such a system would serve 200,000 customers.

The examples in this system are illustrative of the kinds of system one can construct using SMs as the basis building blocks and XCONs as the glue used to combine them easily. Many other variants are possible. In particular, a variety of different NIs is desirable, since not all customers require the same set of services. An NI that supports say 10 Mbs of bandwidth to individual customers can be constructed from a single SM supplemented with small front-end concentrators. Such an NI would require 16 FOLs to a PS and would support 750 customers, making it substantially less expensive than the one described above.

4. Summary

It's worth noting that the network described here can support other connection types in addition to point-to-point and broadcast. First, it is possible to include an upstream capability in broadcast channels. No new mechanisms are required in the SMs. The CP must simply set the LCXT entries so that packets received from broadcast recipients get multiplexed onto the link and logical channel going to the broadcast source. Such an upstream capability can be used in a variety of ways—for example, if the channel is being used to broadcast a lecture or other presentation, the upstream capability gives listeners a way to ask questions.

The presence of an upstream capability raises new questions for congestion control. Since the data rates of all the participants add to the total upstream flow, it's not clear that one can control congestion by policies enforced at the NIs. One approach is to make upstream packets low priority, so that they get through the network only if there is sufficient bandwidth available. This may work adequately in some situation but isn't a general solution.

The system can also support a multi-way broadcast, in which every packet sent by any participant is received by all others. This is a natural sort of connection to use in providing a conference service.

In telephone networks that support conferencing, a new person is added to a conference because of an action initiated by a current participant. In a communications network that supports broadcast television, there is a need for persons not part of a connection to request that they be connected (subject to authorization procedures). Having observed this, one notices that the ability to add on to a connection from 'outside' is a generally useful ability. It allows for example, conferences that users can come and go from as they wish, rather than requiring that they be available when the conference starts. This suggests that every connection should have the potential of adding new

particpants either from inside or outside, with authorization options used to restrict connection from outside if that is desired.

This paper has described the design of a packet switching network that can support voice, data and video communication on a large scale. The most novel aspect of the system is its flexible broadcast capability, which is suitable for a range of services including broadcast television and conferencing. The main purpose of the paper is to show that such systems are technically feasible.

Acknowledgements. This research is an outgrowth of earlier work that I did while at Bell Laboratories. Many individuals contributed ideas to that earlier work, which continues to influence my thinking—particularly, Harold Andrews, Danny Creed, Maurizio Dècina, Bill Hoberecht, Warren Montgomery and Len Wyatt. Thanks are also due to Eric Nussbaum, Pat White and Neil Haller of Bell Communications Research for supporting my continued work in this area and for stimulating my interest in broadcast networks.

REFERENCES

[1] Feng, Tse-yun. "A Survey of Interconnection Networks," *Computer*, vol. 14, no. 12, 12/83, 12–30.

[2] Hoberecht William L. "A Layered Network Protocol for Packet Voice and Data Integration," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1006–1013.

[3] Jenq, Yih-Chyun. "Performance Analysis of a Packet Switch Based on a Single-Buffered Banyan Network," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1014–1021.

[4] Montgomery, Warren A. "Techniques for Packet Voice Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1022–1028.

[5] Turner, Jonathan S. and Leonard F. Wyatt. "A Packet Network Architecture for Integrated Services," *Proceedings of Globecom 83*, 11/83, 45–50.

[6] Turner, Jonathan S. "Packet Load Monitoring By Trunk Controllers," United States Patent #4,484,326, 11/20/84.

[7] Turner, Jonathan S. "Packet Switching Loop-Around Network and Facilities Testing," United States Patent #4,486,877, 12/4/84.

[8] Turner, Jonathan S. "End-to-End Information Memory Arrangement in a Line Controller," United States Patent #4,488,288, 12/11/84.

[9] Turner, Jonathan S. "Interface Facility for a Packet Switching System," United States Patent #4,488,289, 12/11/84.

[10] Turner, Jonathan S. "Packet Error Rate Measurements by Distributed Controllers," United States Patent #4,490,817, 12/25/84.

[11] Turner, Jonathan S. "Fast Packet Switch," United States Patent #4,491,945, 1/1/85.

[12] Turner, Jonathan S. "Fast Packet Switching System," United States Patent #4,494,230, 1/15/85.

[13] Turner, Jonathan S. "Design of an Integrated Services *Packet* Network," Washington University, Computer Science Department, Technical Report WUCS-85-3.